

THE UNIVERSITY OF CHICAGO

Lisa G. Gray

0825141
08/28/97
65723 U.S. PRO

PRONUNCIATION GENERATION IN SPEECH RECOGNITION

Background

The invention relates to generating pronunciations for words added to a dictation vocabulary used in a speech recognition system.

5 A speech recognition system analyzes a person's speech to determine what the person said. Most speech recognition systems are frame-based. In a frame-based system, a processor divides a signal descriptive of the speech to be recognized into a series of digital frames,
10 each of which corresponds to a small time increment of the speech. The processor then compares the digital frames to a set of speech models. Each speech model may represent a word from a vocabulary of words, and may represent how that word is spoken by a variety of speakers. A speech model
15 also may represent a sound, or phoneme, that corresponds to a portion of a word. Collectively, the constituent phonemes for a word represent the phonetic spelling of the word.

The processor determines what the speaker said by finding the speech models that best match the digital frames
20 that represent the person's speech. The words or phrases corresponding to the best matching speech models are referred to as recognition candidates. The processor may produce a single recognition candidate for each utterance, or may produce a list of recognition candidates. Speech
25 recognition is discussed in U.S. Patent No. 4,805,218, entitled "METHOD FOR SPEECH ANALYSIS AND SPEECH RECOGNITION," which is incorporated by reference.

A speech recognition system may be a "discrete" system -- i.e., one which recognizes discrete words or
30 phrases but which requires the speaker to pause briefly between each discrete word or phrase. Alternatively, a

0825141.032297

2025-11-04 14:52:00

speech recognition system may be "continuous" meaning that the recognition software can recognize spoken words or phrases regardless of whether the speaker pauses between them. Continuous speech recognition systems typically have a higher incidence of recognition errors in comparison to discrete recognition systems due to complexities of recognizing continuous speech. A more detailed description of continuous speech recognition is provided in U.S. Patent No. 5,202,952, entitled "LARGE-VOCABULARY CONTINUOUS SPEECH PREFILTERING AND PROCESSING SYSTEM," which is incorporated by reference.

Summary

In one aspect, the invention features adding a word to a speech recognition vocabulary. A spelling and an utterance of the word are received a collection of possible phonetic pronunciations of the word are created. Speech recognition techniques are then used to find a pronunciation from the collection that best matches the utterance of the word. The word is then added to the speech recognition vocabulary using the spelling and the best-matching pronunciation.

The collection of possible phonetic pronunciations of the word is created by comparing the spelling of the word to a rules list of letter strings with associated phonemes. The letter strings of the rules list are searched for a letter string from the spelling of length greater than one letter. The collection of possible phonetic pronunciations is limited to phonetic pronunciations containing phonemes associated with the letter string of length greater than one.

Implementations may further include the following features. The letter strings of the rules list may be searched for a longest length letter string from the

By

spelling. The collection of possible phonetic pronunciations may be limited to phonetic pronunciations containing phonemes associated with the letter string of longest length.

5 The comparing may start with letter strings from the spelling beginning with a first letter of the spelling. The letter strings of the rules list may be searched for an initial longest length letter string from the spelling beginning with the first letter, and then may be searched
10 for a subsequent longest length letter string from the spelling beginning with a letter following the last letter in the initial longest length letter string.

 In another aspect, the invention features adding a word to a speech recognition vocabulary by receiving a
15 spelling and an utterance of the word, and creating a net of possible phonetic pronunciations of the word by comparing the spelling to a rules list of letter strings with associated phonemes. Speech recognition is used to find a pronunciation from the net that best matches the utterance
20 of the word. The word then is added to the speech recognition vocabulary using the spelling and the best-matching pronunciation.

 In another aspect, the invention features assigning a pre-filtering class to a word by matching the first two
25 letters of the word to classified words starting with the same two letters to form a sub-list of classified words. Phonemes of the word are matched to phonemes of the classified words in the sub-list to form a class list. The word then is placed in the class list.

30 Implementations may further include the following features. Phonemes of the word may be matched to phonemes of the classified words by matching the first four phonemes

of the word to the first four phonemes of the classified words.

In another aspect, the invention features assigning a pre-filtering class to a word by performing a direct look-up of a first phoneme of the word in a database of classified words organized alphabetically by their phonemes, matching the first phoneme of the word to a first word in the database having the same first phoneme, selecting the first word in the database having the same first phoneme and a large number (e.g., 199) following words in the database to form a sub-list, matching the first phoneme of the word to the first phoneme of the classified words in the sub-list to form a class list, and placing the word in the class list.

Implementations may further include the following features. The first phoneme of the word may be matched to a first word in the database having the same first phoneme by matching the first four phonemes of the word to a first word in the database having the same first four phonemes. The first word in the database having the same first four phonemes may be selected. The first phoneme of the word may be matched to the first phoneme of the classified words in the sub-list by matching the first four phonemes of the word to the first four phonemes of the classified words in the sub-list.

Other features and advantages will become apparent from the following description, including the drawings, and from the claims.

Brief Description of the Drawings

Fig. 1 is a block diagram of a speech recognition system.

Fig. 19 is a flow chart of another procedure for adding a word to a pre-filter class.

Fig. 20 is a table relating phoneme symbols to human pronunciations.

5

Description

Fig. 1 is a block diagram of a speech recognition system 100. The system includes input/output (I/O) devices (e.g., microphone 105, mouse 110, keyboard 115, and display 120) and a general purpose computer 125 having a processor 130, an I/O unit 135 and a sound card 140. A memory 145 stores data and programs such as an operating system 150, an application program 155 (e.g., a word processing program), and speech recognition software 160.

The microphone 105 receives the user's speech and conveys the speech, in the form of an analog signal, to the sound card 140, which in turn passes the signal through an analog-to-digital (A/D) converter to transform the analog signal into a set of digital samples. Under control of the operating system 150 and the speech recognition software 160, the processor 130 identifies utterances in the user's continuous speech. Utterances are separated from one another by a pause having a sufficiently-large, predetermined duration (e.g., 160-250 milliseconds). Each utterance may include one or more words of the user's speech.

Fig. 2 illustrates components of the speech recognition software 160. For ease of discussion, the following description indicates that the components carry out operations to achieve specified results. However, it should be understood that each component actually causes the processor 130 to operate in the specified manner.

Initially, a front end processing module 200 converts the digital samples 205 from the sound card 140 into frames of parameters 210 that represent the frequency content of an utterance. Each frame includes 24 parameters and represents a short portion (e.g., 10 milliseconds) of the utterance.

As shown in Fig. 3, the front end processing module 200 produces a frame from digital samples according to a procedure 300. The module first produces a frequency domain representation $X(f)$ of the portion of the utterance by performing a Fast Fourier Transform (FFT) on the digital samples (step 305). Next, the module determines $\log(X(f))^2$ (step 310). The module then performs frequency warping (step 315) and a filter bank analysis (step 320) to achieve speaker normalization. See S. Wegmann et al., "Speaker Normalization on Conversational Speech," Proc. 1996 ICASSP, pp. I.339-I.341, which is incorporated by reference.

From the normalized results, the module performs cepstral analysis to produce twelve cepstral parameters (step 325). The module generates the cepstral parameters by performing an inverse cosine transformation on the logarithms of the frequency parameters. Cepstral parameters and cepstral differences have been found to emphasize information important to speech recognition more effectively than do the frequency parameters. After performing channel normalization of the cepstral parameters (step 330), the module produces twelve cepstral differences (i.e., the differences between cepstral parameters in successive frames) (step 335) and twelve cepstral second differences (i.e., the differences between cepstral differences in successive frames) (step 340). Finally, the module performs an IMELDA linear combination transformation to select the twenty four most useful parameters from the twelve cepstral

parameters, the twelve cepstral differences, and the twelve cepstral second differences (step 345).

Referring again to Fig. 2, a recognizer 215 receives and processes the frames of an utterance to identify text corresponding to the utterance. The recognizer entertains several hypotheses about the text and associates a score with each hypothesis. The score reflects the probability that a hypothesis corresponds to the user's speech. For ease of processing, scores are maintained as negative logarithmic values. Accordingly, a lower score indicates a better match (a high probability) while a higher score indicates a less likely match (a lower probability), with the likelihood of the match decreasing as the score increases. After processing the utterance, the recognizer provides the best-scoring hypotheses to the control/interface module 220 as a list of recognition candidates, where each recognition candidate corresponds to a hypothesis and has an associated score. Some recognition candidates may correspond to text while other recognition candidates correspond to commands. Commands may include words, phrases or sentences

The recognizer 215 processes the frames 210 of an utterance in view of one or more constraint grammars 225. A constraint grammar, which also may be referred to as a template or restriction rule, may be a limitation on the words that may correspond to an utterance, a limitation on the order or grammatical form of the words, or both. For example, a constraint grammar for menu-manipulation commands may include only entries from the menu (e.g., "file", "edit") or command words for navigating through the menu (e.g., "up", "down", "top", "bottom"). Different constraint grammars may be active at different times. For example, a constraint grammar may be associated with a particular

application program 155 and may be activated when the user opens the application program and deactivated when the user closes the application program. The recognizer 215 discards any hypothesis that does not comply with an active
5 constraint grammar. In addition, the recognizer 215 may adjust the score of a hypothesis associated with a particular constraint grammar based on characteristics of the constraint grammar.

As shown in Fig. 4A, which illustrates the
10 constraint grammar for a "select" command used to select previously recognized text, a constraint grammar may be illustrated as a state diagram 400. The "select" command includes the word "select" followed by one more previously-recognized words, with the words being in the order
15 recognized. The first state 405 of the constraint grammar indicates that the first word of the select command must be "select". After the word "select", the constraint grammar permits a transition along a path 410 to a second state 415 that requires the next word in the command to be a
20 previously-recognized word. A path 420, which returns to the second state 415, indicates that the command may include additional previously-recognized words. A path 425, which exits the second state 415 and completes the command, indicates that the command may include only previously-
25 recognized words. Fig. 4B illustrates the state diagram 450 of the constraint grammar for the select command when a previously-recognized utterance is "four score and seven". This state diagram could be expanded to include words from additional utterances.

30 The constraint grammar also may be expressed in Backus-Naur Form (BNF) or Extended BNF (EBNF). In EBNF, the grammar for the "select" command is:

<recognition result> ::= <"select"><word>,

where

$$\langle \text{word} \rangle ::= \langle [\text{PRW}^1 [\text{PRW}^2 [\text{PWR3} \dots \text{PRW}^n]]] \mid [\text{PRW}^2 [\text{PWR3} \dots \text{PRW}^n]] \mid \dots \text{PRW}^n \rangle$$

and

5 PRW ::= <previously-recognized word>.

As illustrated in Figs. 4A and 4B, this notation indicates that "select" may be followed by any ordered sequence of previously-recognized words. Constraint grammars are discussed further in U.S. Patent Application No. 08/559,207,
10 filed November 13, 1995 and entitled "CONTINUOUS RECOGNITION OF SPEECH AND COMMANDS", which is incorporated by reference.

One constraint grammar 225 that may be used by the speech recognition software 160 is a large vocabulary dictation grammar. The large vocabulary dictation grammar
15 identifies words included in the active vocabulary 230, which is the vocabulary of words known to the software. The large vocabulary dictation grammar also indicates the frequency with which words occur. The large vocabulary dictation grammar may include a language model. The
20 language model may be a unigram model that indicates the frequency with which a word occurs independently of context, or a bigram model that indicates the frequency with which a word occurs in the context of a preceding word. For example, a bigram model may indicate that a noun or
25 adjective is more likely to follow the word "the" than is a verb or preposition. Other constraint grammars 225 include an in-line dictation macros grammar for dictation commands, such as "/CAP" to capitalize a word and "/New-Paragraph" to start a new paragraph; a select X Y Z grammar used in
30 selecting text, an error correction commands grammar; a dictation editing grammar, an application command and control grammar that may be used to control a particular application program 155; a global command and control

user's speech using an enrollment program. The acoustic models may be further adapted as the system is used. The acoustic models are maintained in a file separate from the active vocabulary 230.

- 5 The acoustic models 235 represent each triphone node as a mixture of Gaussian probability density functions ("PDFs"). For example, node "i" of a triphone "abc" may be represented as ab^ic :

$$ab^ic = \sum_k w_k N(\mu_k, c_k),$$

where each w_k is a mixture weight,

$$\sum_k w_k = 1,$$

- 10 μ_k is a mean vector for the probability density function ("PDF") N_k , and c_k is the covariance matrix for the PDF N_k . Like the frames in the sequence of frames, the vectors μ_k each include 24 parameters. The matrices c_k are twenty four by twenty four matrices. Each triphone node may be
15 represented as a mixture of up to sixteen different PDFs.

- A particular PDF may be used in the representation of multiple triphone nodes. Accordingly, the acoustic models represent each triphone node as a collection of mixture weights w_k associated with up to sixteen different
20 PDFs N_k and separately represent each PDF N_k using a mean vector μ_k and a covariance matrix c_k .

- The recognizer 215 operates in parallel with a pre-filtering procedure 240. Upon initiating processing of an utterance, the recognizer requests from the pre-filtering
25 procedure a list of words that may have been spoken as the first word of the utterance (i.e., words that may correspond to the first and subsequent frames of the utterance). The

00025141.032297

the word corresponding to the node should be added to a word list for a time associated with the node.

After processing the node (step 710), the pre-filtering procedure determines whether the node is the
5 highest node in the tree (i.e., the root node) (step 715).
If the node is not the highest node, then the pre-filtering procedure goes to the next node having no unprocessed active subnodes (step 720) and processes that node (step 710).
When searching for the next node to process, the pre-
10 filtering procedure considers inactive nodes having active subnodes or active siblings.

If the processed node is the highest active node (step 715), then the pre-filtering procedure processes the
silence node 512 (step 725). In general, the silence node
15 is processed by comparing a frame to a model for silence and adding the resulting score to the minimum of the current score for the silence node and the score for the root node 505.

Next, the pre-filtering procedure reseeds the
20 lexical tree (step 730). The pre-filtering procedure reseeds the tree whenever the silence node 512 is active or a word was produced by a leaf node of the lexical tree, regardless of whether the word was added to the list of words. The pre-filtering procedure reseeds the tree by
25 replacing the score for the root node 505 with the minimum of the score for the silence node 512 and the scores for any words produced by leaf nodes of the lexical tree for the current frame. If the silence node is inactive and no leaf node has produced a word, then the pre-filtering procedure
30 replaces the score for the root node 505 with a bad score (i.e., a score having a value larger than a pruning threshold).

A comparison with a frame of parameters may cause the score in a particular state to remain in the state (through a path 820). A score remains in the state when the score, after being adjusted based on a comparison with a model for the state, is better than a score passed from a preceding state or node, or when no score is passed from a preceding state or node. The comparison also may cause the score to be passed to a subsequent state through a path 825. A score is passed to a subsequent state when the score, after being adjusted based on a comparison with a model for the subsequent state, is better than the score in the subsequent state, or when no score is associated with the subsequent state. The score for the third state 815 may be passed to one or more subsequent nodes through a path 830.

Referring to Fig. 8B, the node 512 that corresponds to silence is represented by a single state 840. Each comparison with a frame of parameters may cause a score in the node to remain in the state 840 (through the path 845) and also may cause the score to be passed to the root node 505 through a path 850.

Referring to Fig. 8C, the root node 505 is represented by a single state 860. Comparison with a frame causes the score in the node to be passed to one or more subsequent nodes (including the silence node 512) through a path 865.

Each state of a node may be represented by four values: a score, a starting time, a leaving penalty, and a staying penalty. The score represents the likelihood that a series of frames has placed the lexical tree in the state (i.e., the probability that the series of frames corresponds to the word or portion of a word to which the state corresponds). The scores are maintained as negative logarithmic values.

00025141.032897

The starting time identifies the hypothesized time at which the user began to speak the word or words represented by the state. In particular, the starting time identifies the time at which the score associated with the state entered the lexical tree (i.e., the time at which the score was passed from the state 840 along the path 850).

The leaving and staying penalties are fixed values associated with the state. The staying penalty is added to any score that stays in the state. The staying penalty is related inversely to the length of the sound represented by the state and to the length of the phoneme represented by the node to which the state belongs. For example, the staying penalty could be proportional to $-\log(1 - 1/d_{avg})$, where d_{avg} is the average duration, in frames, of the sound represented by the state. Thus, the staying penalty has a relatively large value when the sound corresponding to the state occurs for only a small amount of time and a relatively small value when the sound corresponding to the state occurs for a large amount of time.

The leaving penalty is added to any score that exits the state, and includes a duration component and a language model component. The duration component is related directly to the length of the sound represented by the state and to the length of the phoneme represented by the node to which the state belongs. For example, the duration component of the leaving penalty could be proportional to $-\log(1/d_{avg})$. Thus, the duration component of the leaving penalty has a relatively large value when the sound corresponding to the state occurs for a large amount of time and a relatively small value when the sound corresponding to the state occurs for a small amount of time.

The language model components of the leaving penalties for all states in a particular node together

After the second frame is retrieved, the score for the state 805 (S_{B2}) is set equal to:

$$S_{B2} = \min(S_{B1} + \text{stay}_B, S_{A1}) + A_{B2},$$

where A_{B2} is the acoustic score resulting from an acoustic match of the second frame with the acoustic model

5 corresponding to state 805 and stay_B is the staying penalty for state 805. The score for state 805 (S_{B2}) corresponds to the more likely of two alternatives: (1) the first frame was silence and the second frame was the sound represented by the state 805 or (2) both of the first and second frames
10 were the sound represented by the state 805. The first alternative corresponds to a transition from state 840 to state 805 along the path 850. The second alternative corresponds to a transition from state 805 back to state 805 along path 820. When the first alternative is the more
15 likely, the starting time corresponding to the first frame that was stored previously for the state 805 is replaced by a value corresponding to the second frame. This value indicates that the score at state 805 represents a word that started with the second frame.

20 After the second frame is retrieved, the state 810 becomes an active state. The score for the state 810 (S_{C2}) is set equal to:

$$S_{C2} = S_{B1} + \text{leave}_B + A_{C2},$$

where A_{C2} is the acoustic score resulting from an acoustic match of the second frame with the acoustic model

25 corresponding to state 810 and leave_B is the leaving penalty for the state 805. Similarly, leave_C and leave_D are leaving penalties for, respectively, states 810 and 815. The sum of language model components of leave_B , leave_C and leave_D

values for each state of the node (step 1105). The node-processing procedure updates the scores and time values by generating acoustic scores and using the equations discussed above.

5 When the last state of the node was active prior to updating the scores for the node, the node-processing procedure uses the score for the last state to generate scores for any inactive subnodes of the node. If the generated score for a subnode does not exceed a pruning
10 threshold, then the node-processing procedure activates that subnode and provides the subnode with the generated score.

 Next, the node-processing procedure determines whether the score of any state of the node exceeds the pruning threshold (step 1110). When a score exceeds the
15 pruning threshold, the likelihood that the word represented by the score was spoken is deemed to be too small to merit further consideration. For this reason, the procedure prunes the lexical tree by deactivating any state having a score that exceeds the pruning threshold (step 1115). If
20 every state of the node is deactivated, then the node-processing procedure also deactivates the node. The node-processing procedure may deactivate a node or state by deleting a record associated with the node or state, or by indicating in the record that the node or state is inactive.
25 Similarly, the node-processing procedure may activate a node or state by creating a record and associating the record with the node or state, or by indicating in an existing record that the node or state is active. The procedure may use a dynamic pruning threshold that accounts for variations
30 in the average or best score in the lexical tree at any given time.

 Next, the node-processing procedure determines whether a word is to be added to a list of words (step

1120). A word is added to the list of words when the node being processed corresponds to the last phoneme of a word, a score has been propagated out of the last state of the node, and the score is less than a list threshold. Before comparing the score to the list threshold, the node-processing procedure adds a language model score to the score. The language model score corresponds to the difference between the language model score for the word and the incremental language model score that is already included in the score. In general, the list threshold has a lower value than the pruning threshold. If the node being processed corresponds to the last phoneme of multiple words having the same phonetic spelling, then all of the words to which the node corresponds are added to the list of words.

If the noted conditions are met, the node-processing procedure adds the word or words to the list (step 1125). A word is stored in the list of words along with the score propagated out of the last state. If the word is on the list already, then the node-processing procedure stores with the list the better of the score already stored with the list or the score propagated out of the last state. The scores for words in a list of words are returned along with the list of words. The recognizer 215 uses these scores in making the detailed match.

The node-processing procedure also adds the word to lists of words for times that precede or follow the starting time to account for possible inaccuracies in the starting time of the word that may result from selecting the better of a score that remains in a state or a score propagated from a prior state. Spreading the word across multiple lists ensures that these inaccuracies will not impinge on the accuracy of the speech recognition system. The node-

25

processing procedure spreads the word across multiple lists based on the length of the word.

After adding a word to the list of words (step 1125), the node-processing procedure saves the score associated with the word as a reseeding score for use in reseeding the tree (step 1130). Production of a word by the lexical tree means that the current frame may correspond to the last frame of the word (with the probability of such a correspondence being reflected by the score associated with the word). This means that the next frame may correspond to the beginning of a word or to silence resulting from a pause between words. The pre-filtering procedure reseeds the tree (step 730 of Fig. 7) to account for this possibility.

For a given frame, multiple nodes may produce words. However, the tree only needs to be reseeded once. To account for this, the node-processing procedure only saves the score associated with a word (S_w) as the reseeding score (S_{RS}) if the word is the first word to be generated by the tree for the current frame or if the word score is less than the score for all other words generated by previously-processed nodes for the current frame (S_{RS}'):

$$S_{RS} = \min(S_w, S_{RS}').$$

Saving only the lowest score (i.e., the score indicating the highest probability that the current frame was the last frame of a word) ensures that the tree will be reseeded using the highest probability that the next frame is the first frame of a new word.

To reseed the tree (step 730 of Fig. 7), the pre-filtering procedure activates the root node 505 and associates the minimum of the reseeding score (S_{RS}) and the score for the silence node 512 with the root node. During

processing of the next frame, the active root node 505 may be used to activate nodes in the group 510 or to activate the silence node 512.

Processing of the node is complete after the node-
5 processing procedure saves a score for use in reseeding the tree (step 1130), or if no word is to be added to the list of words (step 1120). The lexical tree pre-filtering procedure is discussed in detail in U.S. Patent Application No. 08/701,393, filed on August 22, 1996 and entitled
10 "LEXICAL TREE PRE-FILTERING IN SPEECH RECOGNITION", which is incorporated by reference.

After the pre-filtering procedure responds with the requested list of words, the recognizer initiates a hypothesis for each word from the list and compares acoustic
15 models for the word to the frames of parameters representing the utterance. The recognizer uses the results of these comparisons to generate scores for the hypotheses. Hypotheses having excessive scores are eliminated from further consideration. As noted above, hypotheses that
20 comply with no active constraint grammar also are eliminated.

When the recognizer determines that a word of a hypothesis has ended, the recognizer requests from the pre-filtering procedure a list of words that may have been
25 spoken just after the ending-time of the word. The recognizer then generates a new hypotheses for each word on the list, where the new hypothesis includes the words of the old hypothesis plus the new word.

In generating the score for a hypothesis, the
30 recognizer uses acoustic scores for words of the hypothesis, a language model score that indicates the likelihood that words of the hypothesis are used together, and scores provided for each word of the hypothesis by the pre-

to create hypotheses or to expand any hypothesis for which a word has ended (step 1280). Each word in the list of words has an associated score. Prior to adding a list word to a hypothesis, the recognizer modifies the list score (S_L) for
5 the word to produce a modified list score (S_{ML}) as:

$$S_{ML} = S_L + L_C - L_L,$$

where L_C is a language model score that represents the frequency with which the pair of words that includes the list word and the immediately preceding word in the
10 hypothesis are used together in speech, and L_L is a language model score included in the list score and corresponds to the frequency with which the list word is used in speech, without reference to context. The recognizer then adds the modified list score to the score for the hypothesis and
15 compares the result to a threshold value. If the result is less than the threshold value, then the recognizer maintains the hypothesis. Otherwise, the recognizer determines that the hypothesis does not merit further consideration and abandons the hypothesis. As an additional part of creating
20 or expanding the hypotheses, the recognizer compares the hypotheses to the active constraint grammars 225 and abandons any hypothesis that corresponds to no active constraint grammar. The recognizer then retrieves the next frame (step 1210) and repeats the procedure.

25 If there are no more speech frames to process, then the recognizer 215 provides the most likely hypotheses to the control/interface module 220 as recognition candidates (step 1285).

30 The control/interface module 215 controls operation of the speech recognition software and provides an interface to other software or to the user. The control/interface

module receives the list of recognition candidates for each utterance from the recognizer. Recognition candidates may correspond to dictated text, speech recognition commands, or external commands. When the best-scoring recognition

5 candidate corresponds to dictated text, the control/interface module provides the text to an active application, such as a word processor. The control/interface module also may display the best-scoring recognition candidate to the user through a graphical user
10 interface. The control/interface module controls operation of the speech recognition software in response to speech recognition commands (e.g., "wake up", "make that"), and forwards external commands to the appropriate software.

The control/interface module also controls the active vocabulary, acoustic models, and constraint grammars that are used by the recognizer. For example, when the speech recognition software is being used in conjunction with a particular application (e.g., Microsoft Word), the control/interface module updates the active vocabulary to include command words associated with that application and activates constraint grammars associated with the application.

Other functions provided by the control/interface module include an enrollment program, a vocabulary customizer, and a vocabulary manager. The enrollment program collects acoustic information from a user and trains or adapts a user's models based on that information. The vocabulary customizer optimizes the language model of a specific topic by scanning user supplied text. The vocabulary manager is a developer tool which is used to browse and manipulate vocabularies, grammars and macros. Each function of the control/interface module 220 may be

The phoneme list for n=0 and letter string "w" can be expressed as:

xT,0340

	<u>phoneme</u>	<u>probability</u>
5	d@bLyH	100
	hw	670
	v	780
	w	1

Letter counter n is then incremented with its new value equal to its old value plus the length of the letter string in step 1410 (step 1420). The value of letter counter n is checked against the length of the spelled word to see if the end of the word has been reached (step 1425). If the end of the word has not been reached, the longest string of spelled letters in the rules list starting with the letter in the nth place is found (step 1410). In the example, n is incremented to one, and the longest string of spelled letters in the rules list starting with "e", the letter in the first place, is found. The longest letter string is "eigh". The possible phonemes for "eigh", there being two in the rules list, are then stored in the phoneme list with their associated probabilities.

The phoneme list for n=1 and letter string "eigh" can be expressed as:

xT,0341

	<u>phoneme</u>	<u>probability</u>
25	\$	9
	8	239

Letter counter n is then incremented by the length of the letter string "eigh", i.e., n=1+4. The longest string of spelled letters starting with "t" is t (there being no spelled letters after t). The possible phonemes for "t", there being three in the rules list, are then stored in the phoneme list with their associated probabilities.

The phoneme list for n=5 and letter string "t" can be expressed as:

	<u>phoneme</u>	<u>probability</u>
	d	897
5	t	1
	t/	100

When the end of the word has been reached, the constraint grammar is created by forming the complete phonetic spellings of the spelled word from the phonetic fragments in the phoneme list and adding the associated probabilities (step 1430).

	<u>word</u>	<u>probability</u>
	w\$t	11
	w\$t/	110
15	d@bLyH\$t	110
	d@bLyH\$t/	209
	w8t	241
	d@bLyH8t	340
	w8t/	340
20	d@bLyH8t/	439
	etc.	

The constraint grammar is restricted to the 200 phonetic spellings having the highest probabilities (the lowest -log values). Recognizer 215 is then used to recognize the uttered word against the constraint grammar and to select the best phonetic spelling of the spoken word.

Referring to Fig. 15, in the second method 1500 of adding new words to the vocabulary, after the user spells the word (step 1505) and utters the word (step 1510), a rules list is used to create a net containing all possible phonetic spellings of the spelled word (step 1515). See L. Bahl et al., "A Maximum Likelihood Approach to Continuous Speech Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.2, March 1983, pp. 179-190, which is incorporated by reference, for a general

description of net formation. An example of a rules list for use with this method is provided in Appendix B.

Fig. 16 shows an example of a net 1650 created for the word "weight". The rules list provides a list of phonemes 1655 associated with each letter node 1660 of net 1650. Recognizer 215 uses continuous recognition of the uttered word to prune out the paths of net 1650 that do not contain phonemes corresponding to the spoken word to choose the best phonetic spelling (steps 1520 and 1525). The spelled word with the phonetic spelling of the best result is then added to the dictionary (step 1530). In the example of Fig. 16, the paths leading to "gh", "i" and "igh" are pruned out by the recognizer, and "w\$st" is selected by the recognizer. If the acoustic match of the appropriate section of the spoken word to the phonemes 1655 does not result in a strong distinction between possible phonemes 1655, the recognizer will select the phoneme having the higher associated probability.

The phonemes in the rules list of Appendix B can also be used to perform phoneme recognition. Concurrently with using the net, or as a back-off procedure if no match is found using the net, continuous recognition of the uttered word against all of the phonemes in the rules list can be performed. This is particularly effective in cases such as "Fig." pronounced "figure", where the word is abbreviated and the period symbol is not pronounced.

Referring to Fig. 17, to add a new word, the user interfaces with a control window 1750. The word is typed into box 1752, and the users utters the word after turning a microphone on using microphone button 1754. The pronunciation of the word determined by either of the methods described above then appears in the pronunciation

box 1756. The user can edit the pronunciation in box 1756 or directly type the pronunciation into box 1756.

Once the phonetic spelling has been generated it needs to be added to the dictionary. The add button 1758 adds the word to the vocabulary and places the word in a pre-filter class. The new word is assigned a pre-filter class by comparing the new word to words already classified. Referring to Fig. 18, the first two letters of the new word are selected (step 1805) and matched to classified words starting with the same two letters (step 1810) to form a sub-list of classified words (step 1815). For example, if the new word is "hyperbole", a sub-list of classified words is formed that start with "hy". The first four phonemes of the new word are then matched against the first four phonemes of the words in the sub-list (step 1820) to form a class list (step 1825). In the example, the first four phonemes are "h ai p r". The class list contains all classified words in the sub-list that begin with these four phonemes. Words such as "hypertension" are included in the class list. The new word is placed in the class list (step 1830). If there is no match for the first four phonemes, an attempt is made to match the first three, two, or one phonemes, in that order. If no match is found, the search is done on the entire classified word list as a back-off.

Referring to Fig. 19, in a preferred, faster procedure for assigning a pre-filter class to a new word, a direct look-up is performed of the pronunciation of the new word against a database containing a list of all classified words placed in alphabetic order by their first four phonemes (step 1905). The direct look-up matches the first four phonemes of the new word to the first classified word in the list having the same first four phonemes, or a subset of the first four phonemes if there is no match.

here are not limited to any particular hardware or software configuration; they may find applicability in any computing or processing environment that may be used for speech recognition. The techniques may be implemented in hardware or software, or a combination of the two. Preferably, the techniques are implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and one or more output devices. Program code is applied to data entered using the input device to perform the functions described and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (e.g., CD-ROM, hard disk or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described in this document. The system may also be considered to be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

What is claimed is: